Test Plan for MidoNet Fuel Plugin v4.0.0

Revision history

Version	Revision date	Editor	Comment
0.1	23.01.2015	Irina Povolotskaya (ipovolotskaya@mirantis.com)	Created the template structure.
0.2	21.01.2016	Carmela Rubinos (carmela@midokura.com)	Filled template for MidoNet MEM Fuel Plugin for Fuel 7.0
0.3	29.02.2016	Samir Ibradžić (samir@midokura.com)	Add changes related to MidoNet Fuel Plugin v3.0.1, general fixes
0.4	26.04.2016	Samir Ibradžić (samir@midokura.com)	Add changes related to Fuel 8.0 and MidoNet Fuel Plugin v4.0.0

Introduction

MidoNet Fuel Plugin

MidoNet and Midokura Enterprise MidoNet (MEM) are network virtualization software for Infrastructure-as-a-Service (IaaS) clouds, and in the in the OpenStack world they represent an alternative to Neutron's default OpenvSwitch plugin. MidoNet OSS is fully open-source and MEM is the Enterprise, hardened and supported version of our network virtualization software. This Fuel plugin module provides a set of puppet manifests that installs all the components and deploy MidoNet / MEM in a production environment. For a successful MEM deployment, one will also need access to MEM debian package repository credentials.

Developer's specification

MidoNet MEM plugin source code repository [1] contains the developer's specification. MidoNet MEM Fuel Plugin reviews are available in [2].

Limitations

MidoNet / MEM Fuel Plugin version 4.0.0 has been developed for Fuel 8.0 to enable OSS and Midokura Enterprise MidoNet on OpenStack deployments on top of Ubuntu 14.04 platform.

Test strategy

MidoNet / MEM Fuel Plugin replaces Neutron default OpenvSwitch networking plugin in a typical OpenStack deployment based on Neutron. Therefore, the goal of the tests is to to ensure that plugin installation was successful as well to ensure that OpenStack Neutron networking supported by MidoNet is fully functional. As Fuel OSTF health check provide a solid base for testing most of MidoNet functionality, we make sure that all OSTF tests are passing correctly.

Acceptance criteria

The set of Fuel OSTF tests whose successful execution fully depends on MidoNet are:

1. Sanity tests:
    a. Request list of networks
2. Functional tests:
    a. Check network connectivity from instance via floating IP
    b. Check network parameters
    c. Launch instance with file injection

Although the list is somewhat short, executing these functional tests successfully will ensure that a larger set of MidoNet functionality is operating correctly. Just to mention few: MidoNet NSDB (any test listed above), MidoNet REST API (any test listed above), MidoNet Neutron plugin (any test listed above), MidoNet networking agents (tests 2.a and 2.b; on controllers, computes as well as gateways), MidoNet BGP gateway including Floating IPs (tests 2.a and 2.c).

Test environment, infrastructure and tools

Minimum requirement for testing MidoNet / MEM Fuel Plugin in non-HA setup include one OpenStack Controller, one Compute, one MidoNet Gateway and one NSDB node. The base hardware specification of those nodes are:

- CPU: 64-bit x86, quad core or above
- Memory: ≥ 8 GB RAM
- HDD: ≥ 30GB
- NIC: 2 x ≥ 1Gbit

For testing HA capabilities, the minimum needs are 2 Controller nodes, 2 Computes, MidoNet HA Gateway and 3 NSDB. Same base node hardware specification applies. All tests nodes can be both physical hardware as well as the virtual machines (confirmed working on libvirt KVM hypervisor), of same base hardware specification.

Initial tests will be done by simply installing and removing Fuel MidoNet plugin. Next test step is to verify minimal but fully functional deployments with both GRE and VxLAN tunneling, followed by compute scalability test. Finally, full HA scenario test will be done resulting with the following nodes to be deployed and tested:

- 3 NSDB nodes (native HA cluster)
- 3 OpenStack controllers (corosync HA)
- 2 Compute nodes
- 1 Telemetry MongoDB node
- 1 Storage-Cinder node
- 1 MidoNet BGP gateway

An external connectivity tests that verify MidoNet BGP gateway have a special requirement. As of v4.0.0 release of MidoNet Fuel Plugin, the only Neutron network gateway mode supported by the plugin is BGP gateway. That means that the test environment has to have at least one of the external BGP peers available as the endpoint for passing all Floating IP (FIP) traffic between the deployed OpenStack cloud instances and the external network(s). These BGP peers are usually available for production or data-centre ISP environments, so for the sake of supporting BGP tests under lab or proof-of-concept conditions we are providing instructions [3] on how to set up a “fake” BGP peer that supports the OSTF test scenarios that require FIP. To make the testing easier, we assign BGP gateway IP and AS addresses so that they match the values from BGP peer setup example [3]:

- BGP IP subnet: **10.88.88.0/30**
- VyOS BGP peer IP address: **10.88.88.1**
- VyOS BGP peer AS number: **65535**
- MidoNet BGP gateway IP address: **10.88.88.2**
- MidoNet BGP gateway AS number: **12345**
- Floating IP subnet: **200.200.200.0/24**

These parameters can be set in the Neutron MidoNet plugin (OpenStack Settings, “Other” section) via Fuel WEB interface.

Product compatibility matrix

MidoNet plugin is developed for Fuel 8.0 therefore it will be tested against the only supported operating system, Ubuntu 14.04. The supported MidoNet MEM version is v1.9.x (latest 1.9 bug-fix release). For the BGP peer tests, VyOS 1.1.7 or later can be used.

System test scenarios

Install plugin and deploy environment

Test Case ID	**install_plugin_deploy_env**
Steps	1. install plugin using Fuel cli, refer to the installation guide 2. check if plugin is installed successfully, as in the installation guide 3. create environment with enabled plugin in Fuel WEB UI, guide 4. add 1x NSDB, 1x Controller and 1x Compute role node, guide 5. run network verification 6. deploy the cluster 7. check plugin health using cli, as in the installation guide 8. run all OSTF “Sanity tests”
Expected Result	Plugin is installed successfully, All OSTF “Sanity tests” passed, all plugin services are enabled and work as expected. Running time: ~ 2h 20m

Modifying env with enabled plugin (controller scalability)

Test Case ID	**modify_env_with_plugin_remove_add_controller**
Steps	1. install plugin using Fuel cli, refer to the installation guide 2. check if plugin is installed successfully, as in the installation guide 3. create environment with enabled plugin in Fuel WEB UI, guide 4. add 1x NSDB, 3x Controller and 1x Compute role node, guide 5. run network verification 6. deploy the cluster 7. run all OSTF “Sanity tests” 8. remove primary Controller node (usually one with the lowest ID) 9. re-deploy cluster 10. run all OSTF “Sanity tests” 11. add 1x Controller node 12. re-deploy the cluster 13. run all OSTF “Sanity tests”
Expected Result	Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI. Cluster is created and network verification check is passed. Plugin is enabled and configured in the Fuel Web UI. OSTF "Sanity tests" are passing. Environment is deployed successfully. When adding/removing Controller node all plugins resources are migrated to another Controller node. The environment is re-deployed successfully when adding/removing Controller node. Running time: ~3h 30m

Modifying env with enabled plugin (compute scalability)

Test Case ID	**modify_env_with_plugin_remove_add_compute**
Steps	1. install plugin using Fuel cli, refer to the installation guid 2. check if plugin is installed successfully, as in the installation guid 3. create environment with enabled plugin in Fuel WEB UI, guid 4. add 1x NSDB, 1x Controller and 3x Compute role node, guide 5. run network verification 6. deploy the cluster 7. run all OSTF “Sanity tests” 8. remove 1x Compute node 9. re-deploy cluster 10. run all OSTF “Sanity tests” 11. add 1x Compute node 12. re-deploy the cluster 13. run all OSTF “Sanity tests”
Expected Result	Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI. Cluster is created and network check passed. Plugin is enabled in the Fuel Web UI. OSTF "Sanity tests" passing. Environment is deployed successfully. When adding/removing Compute node all plugins resources are migrated to another Compute node. The environment re-deployed successfully when adding/removing Compute node. On second and third run of OSTF “Sanity tests”, the “Check that required services are running” fails, as expected, due to original Compute node being removed. Running time: ~2h

Fuel create mirror and update core repos

Test Case ID	**fuel_create_mirror_update_repos**

Steps	1. install plugin using Fuel cli, refer to the installation guide
	2. check if plugin is installed successfully, as in the installation guide
	3. create environment with enabled plugin in Fuel WEB UI, guide
	4. add 1x NSDB, 1x Controller, 1x Compute and 1x GW role node, guide
	5. run network verification
	6. deploy the cluster
	7. run all OSTF “Sanity tests”
	8. Take note of node IDs and update core repos using Fuel cli:
	`fuel-mirror create -G mos -P ubuntu`
	`fuel --env <env_ID> node --node-id <node_IDs> --tasks setup_repositories`
	9. verify services of all nodes and ensure service PIDs are unchanged
	10. verify if status of all nodes is ‘ready’ using Fuel cli: `fuel nodes`
	11. run all OSTF tests “Sanity tests”

Expected Result	Plugin is installed successfully at the Fuel Master node as indicated in CLI. Cluster is created and network check is passed. Plugin is enabled and configured in the Fuel Web UI. OSTF "Sanity tests" are passing. Environment is deployed successfully. Plugin's services shouldn't be restarted after above tasks were executed. If they are restarted as some exception, this information should be added to plugin's User Guide. Cluster (nodes) should remain in ready state. Re-run of OSTF "Sanity tests" are passing. Running time: ~1h 30m

Uninstall plugin in the deployed environment

Test Case ID	**uninstall_plugin_with_deployed_env**

Steps	1. install plugin using Fuel cli, refer to the installation guide
	2. check that it was installed successfully, as in the installation guide
	3. create environment with enabled plugin in Fuel WEB UI, guide
	4. add 1x NSDB, 1x Controller and 1x Compute role node, guide
	5. run network verification
	6. deploy the cluster
	7. run all OSTF “Sanity tests”
	8. attempt to remove plugin via Fuel CLI:
	`fuel plugins --remove midonet-fuel-plugin==4.0.0`
	9. Ensure that the following output appear as the result in the CLI:
	`400 Client Error: Bad Request (Can't delete plugin which is enabled for some environment.)`

Expected Result	Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI. Cluster is created and network verification check is passed. Plugin is enabled and configured in the Fuel Web UI. OSTF "Sanity tests" are passing. Environment is deployed successfully. Alert is displayed when trying the uninstall the plugin. Running time: ~ 1 h 30 m

Uninstall plugin in the non-deployed environment

Test Case ID	**uninstall_plugin**
Steps	1. install plugin using Fuel cli, refer to the installation guide 2. check that it was installed successfully, as in the installation guide 3. create environment **without** using MidoNet plugin in Fuel WEB UI 4. add 1x Controller and 1x Compute role node, guide 5. run network verification 6. remove plugin via Fuel CLI: `fuel plugins --remove midonet-fuel-plugin==4.0.0`
Expected Result	Plugin is installed successfully at the Fuel Master node and the corresponding output appears in the CLI. Cluster is created and network verification check is passed. Plugin is enabled and configured in the Fuel Web UI. When uninstalling the plugin, no plugin-related elements are left in the environment. Running time: ~ 3 m

MidoNet Full HA

Test Case ID	**midonet_full_ha**
Steps	1. make sure plugin is installed, refer to the installation guide 2. check that it was installed successfully, as in the installation guide 3. create environment with enabled plugin in Fuel WEB UI, guide 4. in addition enable Ceilometer in the Fuel UI 5. add 3x NSDB, 3x Controller, 2x Compute, 1x Telemetry, 1x Storage - Cinder and 1x MidoNet Gateway node 6. deploy the cluster 7. make sure external BGP test peer [3] is set-up and available 8. run all available OSTF test except “Configuration tests”
Expected Result	HA cluster is successfully created, all OSTF tests except “Configuration tests” passed, all plugin services are enabled, stack works correctly. Running time: ~4h !

Appendix

Provide any links to external resources or documentation here.

№	Resources
1	https://github.com/openstack/fuel-plugin-midonet/tree/master
2	https://review.openstack.org/#/q/project:openstack/fuel-plugin-midonet,n,z
3	https://github.com/openstack/fuel-plugin-midonet/blob/master/doc/source/bgp-peer.rst